REMARKS

Applicant is in receipt of the Office Action mailed January 10, 2008. Claims 26-27 have been cancelled. Claims 1-25 and 30 have been amended. Claims 1-25 and 28-30 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Allowed Subject Matter

Applicant appreciates the allowed subject matter of claims 9-13, but believes that the claims as currently amended are patentably distinct and non-obvious over the cited art, as explained below.

Section 101 Rejections

Claims 1-27, 29, and 30 were rejected under 35 U.S.C. 101 as being directed to non-statutory subject matter. Claims 26 and 27 were cancelled, rendering their rejections moot. Claims 1-25 and 30 have been amended to recite a "computer-accessible memory medium" (in place of the non-statutory "medium").

Claim 29 is a means plus function claim. Applicant respectfully submits that such "means plus function" claims are a statutory claim category; *see, e.g.,* MPEP sections 2105-2107, 2181. Applicant further directs the Examiner's attention to 35 U.S.C. Section 112, sixth paragraph, which clearly states that the "means" in means plus function claims specifically refer to structural elements disclosed in the Specification that operate to perform the claimed function. Such means are clearly described in Applicant's Specification, see, e.g., Figures 1A-5 and corresponding text, where various hardware devices, e.g., computers, vision systems, etc., are disclosed for implementing embodiments of the claimed invention.

Thus, Applicant submits that the invention represented in claim 29 is not, and cannot be, "software per se", as asserted by the Examiner, since structural elements *are* included in the claim, per the relevant statute. Moreover, as established in State Street Bank & Trust Company v. Signature Financial Group, Inc., 149 F.3d 1368 (Fed. Cir. 1998), business method and software inventions may certainly be statutory, as long as

they involve a practical application and produce a useful, concrete and tangible result. Applicant respectfully notes that per claim 29, once the task specification has been automatically generated, the function node is operable to execute in accordance with the generated task specification to perform at least a portion of the task, which is certainly a practical application with a tangible concrete result. Applicant thus respectfully submits that the invention as represented in claim 29 meets statutory requirements.

Applicant respectfully submits that claims 1-25, 29, and 30 are directed to statutory subject matter as currently written, and requests removal of the section 101 rejection of these claims.

Section 112 Rejections

Claims 1-30 were rejected under 35 U.S.C. 112, second paragraph, as being indefinite. Claims 26 and 27 were cancelled, rendering their rejections moot. Claims 1-25 and 30 have been amended to recite a "computer-accessible memory medium that stores program instructions" executable to perform the recited functionality (in place of the non-statutory "medium configured…").

Regarding the limitations "initiating a timeout process" and "configuring a timeout event", Applicant respectfully submits that one of skill in the art would readily understand what is meant by these phrases, and that Applicant intends them to be given their common accepted meanings. For example, "initiating a timeout process" refers to a process which is started that is capable of timing out some other process, and "configuring a timeout event" refers to specifying an event in response to which the timeout process will "timeout" or preempt (stop) the executing program. Page 32, lines 6-22 of the present Specification provides further details regarding a preferred embodiment of these features:

> Then, in 624, a timeout process may be initiated. In one embodiment, the timeout process executes at a first priority level. In a preferred embodiment, the timeout process comprises a second execution thread. In one embodiment, initiating the timeout process may be performed by the timed program execution process.

> In 626 a timeout event may be set. Note that the timeout event may be based on a timer, a signal from an external process, and/or any other type of event. For example, a control process or sensor on a conveyor belt or assembly line may send a signal indicating that a next item has arrived, and so indicate that the time period for processing the current item is over, and that the time period for processing the next item is to begin.
>
> In 628, the timed program execution process may perform a time-bounded execution of the program (or a plurality of programs). In one embodiment, the execution of the program may be performed at a second priority level, where the second priority level is below the first priority level. In other words, the process that executes the program preferably operates at a priority level below that of the timeout process, so that the timeout process can reliably preempt the execution process as necessary.

As may be seen, in a preferred embodiment, the timeout process is started (initiated) in a second execution thread that has a higher execution priority than the executing program so that "the timeout process can reliably preempt the execution process as necessary" to interrupt execution of the program (see also, e.g., p.39, lines 17-21). As the above text also explains, the "timeout event", which indicates a timeout condition for the program, can be based on any of a variety of mechanisms, including, for example, a timer, or a signal from an external process (or any other type of event), e.g., a hardware interrupt, among others.

While Applicant believes that the meaning of the claims as originally written is clear, Applicant has amended the independent claims to further clarify the intended meaning of these recited features. Support for these amendments may be found at least in the above-quoted text, as well as in p.27, lines 16–22, p. 27, line 27–p.28, line 10, p.30, lines 7-26, and p.42, line 34–p.44, line 5 of the present Specification.

Applicant thus respectfully submits that as currently written, claims 1-25 and 28-30 are clear, and respectfully requests removal of the section 112 rejection of these claims.

Section 103 Rejections

Claims 1-8, 14-20, and 22-30 were rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen et al. (US Patent No. 6,076,157, "Borkenhagen"), in view of Fuchs et al. (US Patent No. 5,530,802, "Fuchs"). Applicant respectfully disagrees.

Amended Claim 1 recites:

1. A computer-accessible memory medium that stores program instructions for performing time-bounded execution of a program, wherein the program instructions are executable by a processor to perform:

initiating a timed program execution process, wherein the timed program execution process is operable to execute a program in a time-bounded manner;

initiating a timeout process, wherein the timeout process is operable to preempt the execution process to interrupt execution of the program;

configuring a timeout event, wherein the timeout event is an event indicating a timeout condition for the program;

the timed program execution process performing a time-bounded execution of the program, comprising:

a) determining and storing a rollback state for the program;

b) if the timeout event has not occurred, executing the program, wherein, during said executing, if the timeout event occurs,

c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event; and

d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition;

e) performing a program exit procedure;

disabling the timeout event;

terminating the timeout process; and

terminating the timed program execution process.

Applicant respectfully notes that Borkenhagen is directed to forced thread switching, which is very different from Applicant's invention as recited in claim 1. For example, as Borkenhagen makes clear, in Borkenhagen's system, when a thread times out, e.g., due to inactivity ("not performing useful processing"), a thread switch is performed, i.e., switching processing from one thread to another. In direct contrast, in claim 1, the timed program execution process controls the program execution, where the timed program execution process and the timeout process are separate and distinct, e.g., they may execute in different threads and where the timeout process affects the program execution process (which isn't taught by Borkenhagen)–not just switching threads (stopping the process from running), but rather, it changes the program execution process, resetting it to a previous state, specifically, the rollback state, and exiting gracefully with a timeout condition. There are numerous benefits to this approach (which are not provided by the cited art), e.g., in addition to ensuring a graceful exit from the execution process, the rollback state may be specified to ensure a valid state for system resources, e.g., allocated memory, processor mode/state. Fuchs, which is directed to software failure recovery, not time-bounded program execution, also fails to teach or suggest this functionality, as does the combination of Borkenhagen and Fuchs, as will now be explained. Applicant submits that there are numerous features and limitations of claim 1 that are not taught or suggested by the cited art.

For example, nowhere does the cited art disclose **e) performing a program exit procedure** (after rollback), as recited in claim 1.

Cited col.16:22-27 of Borkenhagen reads:

> If, however, the counter value is equal to the threshold value, no thread switch will occur until an instruction can execute, i.e., until forward progress occurs. Note that if the threshold register has value zero, at least one instruction must complete within the active thread before switching to another thread.

As may be seen, this text makes no mention of performing a program exit procedure, but rather describes conditions under which a thread switch may or may not occur. Nor does Fuchs disclose this feature. In fact, the only exiting Fuchs discloses is

that of a "progressive retry recovery algorithm", which is the process that recovers the application, not the application that is rolled back.

Thus, the cited art fails to teach or suggest this claimed feature.

Additionally, nowhere does the cited art disclose: in response to a timeout event during execution, **c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event; and d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition**, as recited in claim 1.

The Office Action admits that Borkenhagen fails to disclose a) determining and storing a rollback state for the program, and (in response to a timeout event during execution) c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event; and d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition, but asserts that Fuchs remedies this admitted deficiency of Borkenhagen, citing col.2:51-60, col.3:24-30, and col.3:30-34. Applicant respectfully disagrees.

For example, cited col.3:24-30 reads:

> In one embodiment, a reorder recovery algorithm is provided that consists only of the receiver reorder step of the overall progressive retry algorithm. When information is known about the particular application process or the cause of the detected fault, it may be determined that the reorder recovery algorithm may be more appropriate for bypassing the fault.

As may be seen, this text in no way teaches or suggests "c) the timeout process setting the timed program execution process to the rollback state, and disabling the timeout event", contrary to the Office Action's assertion. Rather this citation simply discloses provision of a reorder recovery algorithm consisting only of a receiver reorder step in an overall progressive retry algorithm, whose use may be appropriate for bypassing faults in an application process. As Fuchs states in the Abstract, the reorder recovery algorithm attempts "to bypass the detected fault by reordering and reprocessing the inputs that have been received by the faulty application process", and has nothing to do with timed program execution. More specifically, this citation makes no mention of a

timeout process setting a timed program execution process to a rollback state (in response to a timeout event), nor disabling the timeout event.

Applicant further notes that Fuchs in general fails to even mention a timeout process at all, nor a timeout event, and more specifically fails to disclose a timeout process rolling back a timed program execution process to a rollback state in response to a timeout event, and then disabling the timeout event.

Thus, the cited art fails to teach or suggest this claimed feature.

Cited col.3:30-34 reads:

> For example, the reorder recovery algorithm is particularly suitable for bypassing faults where the application process exhibits deterministic behavior, or where the detected fault results from the occurrence of a boundary condition or a racing condition.

Applicant notes that this text makes no mention of "d) the timed program execution process resuming executing the program based on the rollback state with a timeout condition", contrary to the Office Action's assertion. Rather this citation discusses under what conditions "the reorder recovery algorithm is particularly suitable for bypassing faults", e.g., "where the application process exhibits deterministic behavior, or where the detected fault results from the occurrence of a boundary condition or a racing condition". As noted above, per Fuchs's Abstract, the reorder recovery algorithm attempts "to bypass the detected fault by reordering and reprocessing the inputs that have been received by the faulty application process", and has nothing to do with timed program execution. More specifically, this citation makes no mention of a timed program execution process resuming executing a program based on a rollback state with a timeout condition. Moreover, as noted above, Fuchs fails to even mention a timeout process at all, nor a timeout event.

Applicant notes that the timed program execution process resuming executing the program based on the rollback state with a timeout condition allows the program to end gracefully, e.g., without leaving the system or its resources in unstable or inappropriate states, a functionality not provided by Borkenhagen and/or Fuchs.

Thus, the cited art fails to teach or suggest this claimed feature.

Thus, for at least the reasons provided above, Applicant submits that Borkenhagen and Fuchs, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 1.

Applicant further submits that the Office Action has not provided a proper reason to combine Borkenhagen and Fuchs. For example, the only motivation to combine suggested by the Office Action is "because one of the [*sic*]ordinary skills of the art would want to be able to utilize the CPU performance by using the rollback state or any application that has failed or timeout for any event".

Applicant respectfully submits that wanting "to be able to utilize the CPU performance" has no bearing on the novel and beneficial functionality of Applicant's invention as represented by claim 1, at least for the reason that "utilizing the CPU performance" is not germane to resuming executing the program based on the rollback state with a timeout condition (thereby exiting the program execution gracefully). In other words, a primary benefit of the functionality recited in claim 1 is that when a timeout occurs, and the program is rolled back and exits with a timeout condition, the system is prevented from ending up in an undesirable or unpredictable state, which is not really germane to "utilizing CPU performance". Thus, Applicant submits that Borkenhagen and Fuchs are not properly combinable to make a prima facie case of obviousness.

Moreover, even were Borkenhagen and Fuchs properly combinable, which Applicant argues they are not, the resulting combination would still not produce Applicant's invention as recited in claim 1, as discussed above at length.

Thus, for at least the reasons provided above, Applicant submits that claim 1, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 28, 29, and 30 include similar novel limitations as claim 1, and so the above arguments apply with equal force to these claims. Thus, for at least the reasons provided above, Applicant submits that claims 28, 29, and 30, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 21 was rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen in view of Fuchs, and further in view of Chamberlain (US Patent No. 6,438,749).

Applicant respectfully submits that since claim 21 depends from independent claim 1, which was shown above to be patentably distinct and non-obvious, and thus allowable, claim 21 is similarly patentably distinct and non-obvious, and thus allowable, for at least the reasons provided above.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Removal of the section 103 rejection of claims 1-8, 14-21, and 22-30 is earnestly requested.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-81401/JCH.

Also filed herewith are the following items:

☐ Request for Continued Examination

☐ Terminal Disclaimer

☐ Power of Attorney By Assignee and Revocation of Previous Powers

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/

Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 2008-04-08 JCH/MSW